```
from google.colab import files
f=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving heart.csv to heart.csv
```

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import pandas as pd
import numpy as np
df=pd.read_csv('heart.csv')
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

```
df.shape
```

```
(303, 14)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

The variables types are

- Binary: sex, fbs, exang, target
- Categorical: cp, restecg, slope, ca, thal

- Continuous: age, trestbps, chol, thalac, oldpeak

```
df.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

```
# to know unique values
df.nunique()
```

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

```
# change the categorical type to categorical variables
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
df.dtypes
```

```
age           int64
sex          object
cp           object
trestbps      int64
chol          int64
fbs          object
restecg      object
thalach       int64
exang        object
oldpeak     float64
slope        object
ca           object
thal         object
target        int64
dtype: object
```

Error Correction

Check for the data characters mistakes

feature ‘ca’ ranges from 0–3, however, df.nunique() listed 0–4. So lets find the ‘4’ and change them to NaN.

```
df['ca'].unique()
```

```
array([0, 2, 1, 3, 4], dtype=object)
```

```
# to count the number in of each category decending order
df.ca.value_counts()
```

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

```
df[df['ca']==4]
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	s
92	52	1	2	138	223	0	1	169	0	0.0	
158	58	1	1	125	220	0	1	144	0	0.4	
163	38	1	2	138	175	0	1	173	0	0.0	
164	38	1	2	138	175	0	1	173	0	0.0	

```
df.loc[df['ca']==4,'ca']=np.NaN
```

```
df['ca'].unique()
```

```
array([0, 2, 1, 3, nan], dtype=object)
```

Feature ‘thal’ ranges from 1–3, however, df.nunique() listed 0–3. There are two values of ‘0’. So lets change them to NaN

```
df.thal.value_counts()
```

```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

```
df.loc[df['thal']==0,'thal']=np.NaN
```

```
df[df['thal']==0]
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo

```
df['thal'].unique()
```

```
array([1, 2, 3, nan], dtype=object)
```

italicized text###Check for missing values and replace them

```
df.isna().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          5
thal        2
```

```
target      0
dtype: int64
```

```
df = df.fillna(df.median())
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Check for duplicate rows

```
duplicated=df.duplicated().sum()
if duplicated:
  print("Duplicated rows :{}".format(duplicated))
else:
  print("No duplicates")
```

```
Duplicated rows :1
```

```
duplicates=df[df.duplicated(keep=False)]
duplicates.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	s
163	38	1	2	138	175	0	1	173	0	0.0	

statistical summary

1. check on the min and max value for the categorical variables (min-max). Sex (0–1), cp (0–3), fbs (0–1), restecg (0–2), exang (0–1), slope (0–2), ca (0–3), thal (0–3).
2. Observe the mean, std, 25% and 75% on the continuous variables.

```
df.describe()
```

	age	sex	cp	trestbps	chol	fb
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.14851
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.35619
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.00000

Before we plot the outliers, let's change the labeling for better visualization and interpretation.

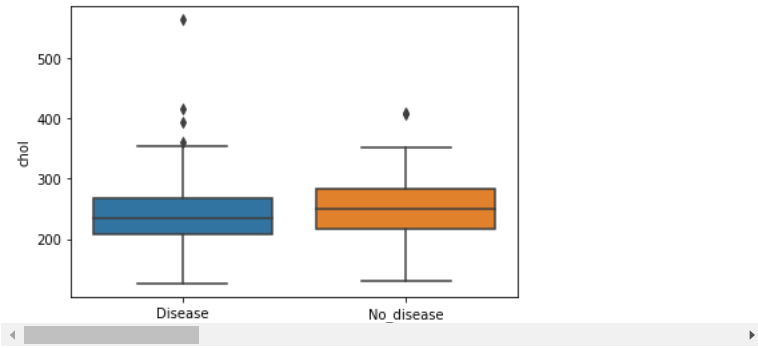
```
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
```

```
df['cp'] = df.cp.replace({0: "typical_angina",
                          1: "atypical_angina",
                          2:"non-anginal pain",
                          3: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slope'] = df.slope.replace({0: "upsloping", 1: "flat",2:"downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect", 3:"normal"})
```

Outliers Detection & Handling

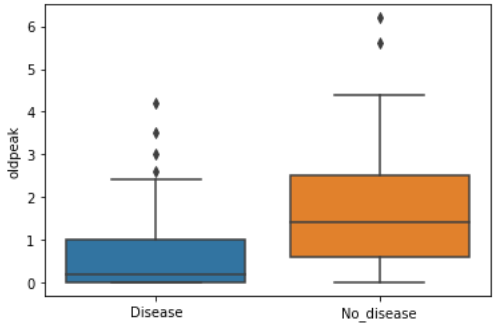
```
import matplotlib.pyplot as plt
import seaborn as sb
bxplt = sb.boxplot(df["target"],df["chol"])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
  FutureWarning
```




```
sb.boxplot(x='target', y='oldpeak', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7715c1cf10>
```




```
# define continuous variable & plot
continous_features = ['age','trestbps','chol','thalach','oldpeak']
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.) # 25th percentile of the data of the given feature
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of the given feature
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) & (feature_data <= Q3 + outlier_step))].index.tolist()
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_feature, len(outliers)))
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature))

outliers(df[continous_features])
```

```
For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 9
For the feature chol, No of Outliers is 5
```

```
For the feature thalach, No of Outliers is 1
For the feature oldpeak, No of Outliers is 5
```

Drop Outliers

```
outliers(df[continous_features],drop=True)
```

```
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed
```

```
from sklearn import preprocessing
df=df.apply(preprocessing.LabelEncoder().fit_transform)
```

```
X = df.drop('target', axis=1)
X.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slc
0	29	1	0	30	62	1	0	47	0	22	
1	3	1	2	21	77	0	1	82	0	32	
2	7	0	1	21	34	0	0	69	0	14	
3	22	1	1	13	65	0	1	74	0	8	

```
X.shape
```

```
(284, 13)
```

```
y = df['target']
```

```
y.head(10)
```

```
0     Disease
1     Disease
2     Disease
3     Disease
4     Disease
5     Disease
6     Disease
7     Disease
9     Disease
10    Disease
Name: target, dtype: object
```

```
y.shape
```

```
(284,)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
print("X_train : ",X_train.shape)
print("X_test : ",X_test.shape)
print("y_train : ",y_train.shape)
print("y_test : ",y_test.shape)
```

```
X_train :  (227, 13)
X_test :  (57, 13)
y_train :  (227,)
y_test :  (57,)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
print(y_pred)
```

```
[0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1
 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1]
```